The following Listing of Claims will replace all prior versions, and listings, of claims in the present application:

Listing of Claims:

1. through 8. **(Cancelled)**

9. **(Currently amended)** A computer ~~implemented~~ system for processing privileges, comprising:

a kernel for enforcing a security policy on a plurality of processes based on privileges;

a memory for storing a privilege model; and

a central processor unit for executing the privilege model stored in the memory, the [[a]] privilege model interfacing with said kernel and implementing a framework in which super-user based processes of said plurality of processes and privilege based processes of said plurality of processes transparently interface with said kernel, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes; and

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; ~~and~~

wherein the framework includes a software module for automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

10. (Original) A system as described in Claim 9 wherein said plurality of privilege sets associated with each process comprises:

an effective set indicating privileges in effect for said process;

a permitted set indicating privileges that can be made effective; and

a limit set indicating an upper bound on all effective sets.

11. (Original) A system as described in Claim 9 wherein each process comprises:

an effective user identification;

a real user identification; and

a saved user identification.

12. (Original) A system as described in Claim 10 wherein each process comprises:

an effective user identification;

a real user identification; and

a saved user identification.

13. (Original) A system as described in Claim 9 wherein said software module automatically updates a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

14. (Original) A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

15. (Original) A system as described in Claim 14 wherein said software module automatically modifies said plurality of privilege sets for said privilege unaware super-user based process according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

16. (Original) A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process remains at an initial state.

17. (Original) A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege aware super-user based process transitioning to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process.

18. (Original) A system as described in Claim 12 wherein said software module does not alter any of a plurality of privilege sets of a privilege aware super-user based process in response to changes in any of its user identification values.

19. (Original) A system as described in Claim 12 wherein said software module allows a privilege aware super-user based process to transition to a privilege

unaware super-user based process without restrictions provided all its user identification values are non-zero.

20. (Original) A system as described in Claim 12 wherein a process of said plurality of processes can directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added or subtracted from a limit set of said process.

21. (Original) A system as described in Claim 10 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlay a first process.

22. (Original) A method of processing privileges comprising:

enforcing a security policy on a plurality of processes based on privileges, said enforcing performed by a kernel of an operating system; and

transparently interfacing super-user based processes of said plurality of processes and privilege based processes of said plurality of processes with said kernel using a privilege model as an intermediary, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes; and

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; and

wherein said transparently interfacing further comprises automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

23. (Original) A method as described in Claim 22 wherein said plurality of privilege sets associated with each process comprises:

an effective set indicating privileges in effect for said process;

a permitted set indicating privileges that can be made effective; and

a limit set indicating an upper bound on all effective sets.

24. (Original) A method as described in Claim 22 wherein each process comprises:

an effective user identification;

a real user identification; and

a saved user identification.

25. (Original) A method as described in Claim 23 wherein each process comprises:

an effective user identification;

a real user identification; and

a saved user identification.

26. (Original) A method as described in Claim 22 wherein said automatically modifying further comprises automatic updating of a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

27. (Original) A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

28. (Original) A method as described in Claim 27 wherein said automatically modifying a plurality of privilege sets for a privilege unaware super-user based process is performed according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

29. (Original) A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process reverts remains at initial state.

30. (Original) A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege aware super-user based process transitioning to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process.

31. (Original) A method as described in Claim 25 wherein said automatically modifying does not alter any of a plurality of privilege sets of a privilege aware super-user based process in response to changes in any of its user identification values.

32. (Original) A method as described in Claim 25 wherein said transparently interfacing further comprises allowing a privilege aware super-user based process to transition to a privilege unaware super-user based process without restrictions provided all its user identification values are non-zero.

33. (Original) A method as described in Claim 25 wherein said transparently interfacing further comprises allowing a process of said plurality of processes to directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added or subtracted from a limit set of said process.

34. (Original) A method as described in Claim 23 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlays a first process.

35. **(Currently amended)** A computer system for processing privileges, comprising:

a kernel for enforcing a security policy on a plurality of processes based on privileges;

a memory for storing a privilege model; and a central processor unit for executing the privilege model stored in the memory, the [[a]] privilege model implementing a framework configured for transparently interfacing super-user based processes of said plurality of processes and privilege based processes of said plurality of processes ~~transparently interface~~ with said kernel, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes, wherein said plurality of privilege sets comprises:

an effective set indicating privileges in effect;

a permitted set indicating privileges that can be made effective; and

a limit set indicating an upper bound on all effective sets;

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; ~~and~~

[[a]] wherein the framework includes a software module for automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

36. (Original) A system as described in Claim 35 wherein each process comprises:

an effective user identification;

a real user identification; and

a saved user identification.

37. (Original) A system as described in Claim 36 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlays a first process.

38. (Original) A system as described in Claim 35 wherein said software module automatically updates a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

39. (Original) A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

40. (Original) A system as described in Claim 39 wherein said software module automatically modifies said plurality of privilege sets for said privilege unaware super-user based process according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

41. (Original) A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process remains at an initial state.

42. (Original) A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege aware super-user based process transitioning to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process.

43. (Original) A system as described in Claim 36 wherein said software module does not alter any of a plurality sets of a privilege aware super-user based process in response to changes in any of its user identification values.

44. (Original) A system as described in Claim 36 wherein said software module allows a privilege aware super-user based process to transition to a privilege unaware super-user based process without restrictions provided all its user identification values are non-zero.

45. (Original) A system as described in Claim 36 wherein a process of said plurality of processes can directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added to or subtracted from a limit set of said process.